PCT

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(54) Title: A WAY PREDICTION STRUCTURE

(57) Abstract

A way prediction structure is provided which predicts a way of an associative cache in which an access will hit, and causes the data bytes from the predicted way to be conveyed as the output of the cache. The typical tag comparisons to the request address are bypassed for data byte selection, causing the access time of the associative cache to be substantially the access time of the direct–mapped way prediction array within the way prediction structure. Also included in the way prediction structure is a way prediction control unit configured to update the way prediction array when an incorrect way prediction is detected. The clock cycle of a superscalar microprocessor including the way prediction structure with its caches may be increased if the cache access time is limiting the clock cycle. Additionally, the associative cache may be retained in the high frequency superscalar microprocessor (which might otherwise employ a direct–mapped cache for access time reasons). Single clock cycle cache access to an associative data cache is maintained for high frequency operation.

TITLE: A WAY PREDICTION STRUCTURE

5   BACKGROUND OF THE INVENTION

1.   Field of the Invention

        This invention is related to the field of superscalar microprocessors and, more particularly, to way
10   prediction structures employed within superscalar microprocessors in order to allow associative cache
        accesses at speeds comparable to direct-mapped cache accesses.

2.   Description of the Relevant Art

15   Superscalar microprocessors achieve high performance by simultaneously executing multiple
        instructions in a clock cycle and by specifying the shortest possible clock cycle consistent with the design.
        As used herein, the term "clock cycle" refers to an interval of time during which the pipeline stages of a
        microprocessor perform their intended functions. At the end of a clock cycle, the resulting values are moved
        to the next pipeline stage.
20
        Since superscalar microprocessors execute multiple instructions per clock cycle and the clock cycle
        is short, a high bandwidth memory system is required to provide instructions and data to the superscalar
        microprocessor (i.e. a memory system that can provide a large number of bytes in a short period of time).
        Without a high bandwidth memory system, the microprocessor would spend a large number of clock cycles
25   waiting for instructions or data to be provided, then would execute the received instructions and/or the
        instructions dependent upon the received data in a relatively small number of clock cycles. Overall
        performance would be degraded by the large number of idle clock cycles. However, superscalar
        microprocessors are ordinarily configured into computer systems with a large main memory composed of
        dynamic random access memory (DRAM) cells. DRAM cells are characterized by access times which are
30   significantly longer than the clock cycle of modern superscalar microprocessors. Also, DRAM cells typically
        provide a relatively narrow output bus to convey the stored bytes to the superscalar microprocessor.
        Therefore, DRAM cells provide a memory system that provides a relatively small number of bytes in a
        relatively long period of time, and do not form a high bandwidth memory system.

35   Because superscalar microprocessors are typically not configured into a computer system with a
        memory system having sufficient bandwidth to continuously provide instructions and data, superscalar
        microprocessors are often configured with caches. Caches are storage devices containing multiple blocks of
        storage locations, configured on the same silicon substrate as the microprocessor or coupled nearby. The
        blocks of storage locations are used to hold previously fetched instruction or data bytes. The bytes can be
40   transferred from the cache to the destination (a register or an instruction processing pipeline) quickly;

                                                   1

commonly one or two clock cycles are required as opposed to a large number of clock cycles to transfer
bytes from a DRAM main memory.

Caches may be organized into an "associative" structure (also referred to as "set associative"). In an
5   associative structure, the blocks of storage locations are accessed as a two-dimensional array having rows
(often referred to as "sets") and columns (often referred to as "ways"). When a cache is searched for bytes
residing at an address, a number of bits from the address are used as an "index" into the cache. The index
selects a particular set within the two-dimensional array, and therefore the number of address bits required
for the index is determined by the number of sets configured into the cache. The act of selecting a set via an
10   index is referred to as "indexing". The addresses associated with bytes stored in the multiple blocks of a set
are examined to determine if any of the addresses stored in the set match the requested address. If a match is
found, the access is said to be a "hit", and the cache provides the associated bytes. If a match is not found,
the access is said to be a "miss". When a miss is detected, the bytes are transferred from the memory system
into the cache. The addresses associated with bytes stored in the cache are also stored. These stored
15   addresses are referred to as "tags" or "tag addresses".

The blocks of memory configured into a set form the columns of the set. Each block of memory is
referred to as a "way"; multiple ways comprise a set. The way is selected by providing a way value to the
cache. The way value is determined by examining the tags for a set and finding a match between one of the
20   tags and the requested address. A cache designed with one way per set is referred to as a "direct-mapped
cache". In a direct-mapped cache, the tag must be examined to determine if an access is a hit, but the tag
examination is not required to select which bytes are transferred to the outputs of the cache. Since only an
index is required to select bytes from a direct-mapped cache, the direct-mapped cache is a "linear array"
requiring only a single value to select a storage location within it.

25

Both direct-mapped and associative caches are employed in high frequency (i.e. short clock cycle)
superscalar microprocessors. In high frequency applications, set associative caches either become a clock
cycle limiter or require multiple clock cycles to complete a cache access because the comparison of tags to
the request address and the subsequent selection of data bytes to convey to the output requires more time than
30   the desired clock cycle time allows. Cache bandwidth (i.e. the number of cache requests which may be made
per clock cycle) decrease due to multiple clock cycle accesses is particularly detrimental to superscalar
microprocessors attempting to execute multiple instructions (which may require cache access) per clock
cycle. Although cache bandwidth decrease may be avoided by pipelining the cache access, the latency for
each access is still detrimental to performance. Longer clock cycles are also detrimental to the performance
35   of a microprocessor, since fewer instructions may be executed per second.

Direct-mapped caches, which compare the selected tag to the request address in parallel with
conveying data bytes to the output, operate in less time than the associative cache. Unfortunately, direct-
mapped caches are associated with lower hit rates (i.e. the percentage of accesses that are hits) than

associative caches with a similar storage capacity. Furthermore, direct-mapped caches are more susceptible to "thrashing". Thrashing is a phenomenon that occurs when the pattern of address requests presented to the cache contains several dissimilar addresses with the same index. Dissimilar addresses are addresses that are stored in the cache with different tags. As an illustrative example, addresses A and B may access the cache
5   alternately and repeatedly. Address A and address B have the same index, and access a direct-mapped cache. First, address A accesses the cache and misses. The indexed cache storage location is filled with bytes associated with address A. Next, address B accesses the cache and misses. The indexed cache storage location discards the bytes associated with address A and is filled with bytes associated with address B. Address A accesses the cache again, and misses. The cache storage location discards the bytes associated
10   with address B and is filled with bytes associated with address A. An associative cache would be able to store bytes associated with both address A and address B simultaneously. A storage device having the access time of a direct-mapped cache with the hit rate and insensitivity to thrashing of an associative cache is desired.

15   SUMMARY OF THE INVENTION

       The problems outlined above are in large part solved by a way prediction structure according to the present invention. The present way prediction structure includes a way prediction array and a way prediction control unit. The way prediction structure described herein predicts a way of an associative data cache in
20   which an access will hit, and causes the data bytes from the predicted way to be conveyed as the output of the data cache. The typical tag comparisons to the input request address are bypassed for data byte selection. Advantageously, the access time of the associative cache is substantially the access time of the direct-mapped way prediction array. The present associative cache and way prediction structure enable increased performance in a high frequency superscalar microprocessor in several ways. First, the clock cycle of the
25   superscalar microprocessor may be increased if the cache access time is limiting the clock cycle (as is typically the case), since the cache access time of the associative cache is shortened. Second, the associative cache is retained in the high frequency superscalar microprocessor (which might otherwise employ a direct-mapped cache for access time reasons). The advantages of an associative cache over a direct-mapped cache, including insensitivity to thrashing and superior hit rate characteristics, are thereby available at higher
30   frequencies then were previously attainable.

       As a result of the shortened access time enabled by the present invention, single clock cycle cache access to an associative data cache is maintained for high frequency operation. Since most operations complete cache access in a single cycle, the cache bandwidth (as measured in accesses per clock cycle) is
35   higher than a cache in which accesses require multiple cycles. The increased cache bandwidth is advantageously available for additional cache accesses. Additionally, performance may be advantageously increased by the reduced latency associated with performing the access in a single clock cycle. Cache bandwidth increase and latency decrease are particularly valuable to superscalar microprocessors in which multiple instructions (possibly requiring cache access) are executed in a given clock cycle.

3

Broadly speaking, the present invention contemplates a way prediction structure for a cache comprising a way prediction array and a way prediction control unit. The way prediction array includes a plurality of storage locations wherein each one of the plurality of storage locations is configured to store a

5   way value. The way prediction array is configured to receive at least one request address. Coupled to the way prediction array, the way prediction control unit is configured to update the way prediction array.

BRIEF DESCRIPTION OF THE DRAWINGS

10

Other objects and advantages of the invention will become apparent upon reading the following detailed description and upon reference to the accompanying drawings in which:

Figure 1 is a block diagram of a superscalar microprocessor including a data cache having a way

15   prediction structure according to the present invention.

Figure 2 is a block diagram of the data cache shown in Figure 1, depicting several portions of one embodiment of the way prediction structure according to the present invention.

20   Figure 3 is a timing diagram depicting a sequence of two cache accesses in which the way prediction is correct.

Figure 4 is a timing diagram depicting a sequence of two cache accesses in which the way prediction for the first cache access is incorrect.

25

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications,

30   equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF THE INVENTION

35

Referring next to Figure 1, a block diagram of a superscalar microprocessor 200 including a data cache 224 in accordance with the present invention is shown. As illustrated in the embodiment of Figure 1, superscalar microprocessor 200 includes a prefetch/predecode unit 202 and a branch prediction unit 220 coupled to an instruction cache 204. Instruction alignment unit 206 is coupled between instruction cache 204

and a plurality of decode units 208A-208D (referred to collectively as decode units 208). Each decode unit 208A-208D is coupled to respective reservation station units 210A-210D (referred to collectively as reservation stations 210), and each reservation station 210A-210D is coupled to a respective functional unit 212A-212D (referred to collectively as functional units 212). Decode units 208, reservation stations 210,

5   and functional units 212 are further coupled to a reorder buffer 216, a register file 218 and a load/store unit 222. A data cache 224 is finally shown coupled to load/store unit 222, and an MROM unit 209 is shown coupled to instruction alignment unit 206.

        Generally speaking, instruction cache 204 is a high speed cache memory provided to temporarily

10   store instructions prior to their dispatch to decode units 208. In one embodiment, instruction cache 204 is configured to cache up to 32 kilobytes of instruction code organized in lines of 16 bytes each (where each byte consists of 8 bits). During operation, instruction code is provided to instruction cache 204 by prefetching code from a main memory (not shown) through prefetch/predecode unit 202. It is noted that instruction cache 204 could be implemented in a set-associative, a fully-associative, or a direct-mapped

15   configuration.

        Prefetch/predecode unit 202 is provided to prefetch instruction code from the main memory for storage within instruction cache 204. In one embodiment, prefetch/predecode unit 202 is configured to burst 64-bit wide code from the main memory into instruction cache 204. It is understood that a variety of specific

20   code prefetching techniques and algorithms may be employed by prefetch/predecode unit 202.

        As prefetch/predecode unit 202 fetches instructions from the main memory, it generates three predecode bits associated with each byte of instruction code: a start bit, an end bit, and a "functional" bit. The predecode bits form tags indicative of the boundaries of each instruction. The predecode tags may also

25   convey additional information such as whether a given instruction can be decoded directly by decode units 208 or whether the instruction must be executed by invoking a microcode procedure controlled by MROM unit 209, as will be described in greater detail below.

        Table 1 indicates one encoding of the predecode tags. As indicated within the table, if a given byte

30   is the first byte of an instruction, the start bit for that byte is set. If the byte is the last byte of an instruction, the end bit for that byte is set. If a particular instruction cannot be directly decoded by the decode units 208, the functional bit associated with the first byte of the instruction is set. On the other hand, if the instruction can be directly decoded by the decode units 208, the functional bit associated with the first byte of the instruction is cleared.   The functional bit for the second byte of a particular instruction is cleared if the

35   opcode is the first byte, and is set if the opcode is the second byte. It is noted that in situations where the opcode is the second byte, the first byte is a prefix byte. The functional bit values for instruction byte numbers 3-8 indicate whether the byte is a MODRM or an SIB byte, or whether the byte contains displacement or immediate data.

Table 1. Encoding of Start, End and Functional Bits

Instr. Byte Number	Start Bit Value	End Bit Value	Functional Bit Value	Meaning
1	1	X	0	Fast decode
1	1	X	1	MROM instr.
2	0	X	0	Opcode is first byte
2	0	X	1	Opcode is this byte, first byte is prefix
3-8	0	X	0	Mod R/M or SIB byte
3-8	0	X	1	Displacement or immediate data; the second functional bit set in bytes 3-8 indicates immediate data
1-8	X	0	X	Not last byte of instruction
1-8	X	1	X	Last byte of instruction

As stated previously, in one embodiment certain instructions within the x86 instruction set may be directly decoded by decode unit 208. These instructions are referred to as "fast path" instructions. The remaining instructions of the x86 instruction set are referred to as "MROM instructions". MROM instructions are executed by invoking MROM unit 209. More specifically, when an MROM instruction is encountered, MROM unit 209 parses and serializes the instruction into a subset of defined fast path instructions to effectuate a desired operation. A listing of exemplary x86 instructions categorized as fast path instructions as well as a description of the manner of handling both fast path and MROM instructions will be provided further below.

Instruction alignment unit 206 is provided to channel variable byte length instructions from instruction cache 204 to fixed issue positions formed by decode units 208A-208D. Instruction alignment unit 206 independently and in parallel selects instructions from three groups of instruction bytes provided by instruction cache 204 and arranges these bytes into three groups of preliminary issue positions. Each group of issue positions is associated with one of the three groups of instruction bytes. The preliminary issue positions are then merged together to form the final issue positions, each of which is coupled to one of decode units 208.

Before proceeding with a detailed description of the data cache 224, general aspects regarding other subsystems employed within the exemplary superscalar microprocessor 200 of Figure 1 will be described. For the embodiment of Figure 1, each of the decode units 208 includes decoding circuitry for decoding the predetermined fast path instructions referred to above. In addition, each decode unit 208A-208D routes displacement and immediate data to a corresponding reservation station unit 210A-210D. Output signals from the decode units 208 include bit-encoded execution instructions for the functional units 212 as well as operand address information, immediate data and/or displacement data.

The superscalar microprocessor of Figure 1 supports out of order execution, and thus employs

reorder buffer 216 to keep track of the original program sequence for register read and write operations, to implement register renaming, to allow for speculative instruction execution and branch misprediction recovery, and to facilitate precise exceptions. As will be appreciated by those of skill in the art, a temporary storage location within reorder buffer 216 is reserved upon decode of an instruction that involves the update

5   of a register to thereby store speculative register states. Reorder buffer 216 may be implemented in a first-in-first-out configuration wherein speculative results move to the "bottom" of the buffer as they are validated and written to the register file, thus making room for new entries at the "top" of the buffer. Other specific configurations of reorder buffer 216 are also possible, as will be described further below. If a branch prediction is incorrect, the results of speculatively-executed instructions along the mispredicted path can be

10   invalidated in the buffer before they are written to register file 218.

The bit-encoded execution instructions and immediate data provided at the outputs of decode units 208A-208D are routed directly to respective reservation station units 210A-210D. In one embodiment, each reservation station unit 210A-210D is capable of holding instruction information (i.e., bit encoded execution

15   bits as well as operand values, operand tags and/or immediate data) for up to three pending instructions awaiting issue to the corresponding functional unit. It is noted that for the embodiment of Figure 1, each decode unit 208A-208D is associated with a dedicated reservation station unit 210A-210D, and that each reservation station unit 210A-210D is similarly associated with a dedicated functional unit 212A-212D. Accordingly, four dedicated "issue positions" are formed by decode units 208, reservation station units 210

20   and functional units 212. Instructions aligned and dispatched to issue position 0 through decode unit 208A are passed to reservation station unit 210A and subsequently to functional unit 212A for execution. Similarly, instructions aligned and dispatched to decode unit 208B are passed to reservation station unit 210B and into functional unit 212B, and so on.

25   Upon decode of a particular instruction, if a required operand is a register location, register address information is routed to reorder buffer 216 and register file 218 simultaneously. Those of skill in the art will appreciate that the x86 register file includes eight 32 bit real registers (i.e., typically referred to as EAX, EBX, ECX, EDX, EBP, ESI, EDI and ESP). Reorder buffer 216 contains temporary storage locations for results which change the contents of these registers to thereby allow out of order execution. A temporary

30   storage location of reorder buffer 216 is reserved for each instruction which, upon decode, is determined to modify the contents of one of the real registers. Therefore, at various points during execution of a particular program, reorder buffer 216 may have one or more locations which contain the speculatively executed contents of a given register. If following decode of a given instruction it is determined that reorder buffer 216 has a previous location or locations assigned to a register used as an operand in the given instruction, the

35   reorder buffer 216 forwards to the corresponding reservation station either: 1) the value in the most recently assigned location, or 2) a tag for the most recently assigned location if the value has not yet been produced by the functional unit that will eventually execute the previous instruction. If the reorder buffer has a location reserved for a given register, the operand value (or tag) is provided from reorder buffer 216 rather than from register file 218. If there is no location reserved for a required register in reorder buffer 216, the

value is taken directly from register file 218. If the operand corresponds to a memory location, the operand value is provided to the reservation station unit through load/store unit 222.

Details regarding suitable reorder buffer implementations may be found within the publication "Superscalar Microprocessor Design" by Mike Johnson, Prentice-Hall, Englewood Cliffs, New Jersey, 1991, and within the co-pending, commonly assigned patent application entitled "High Performance Superscalar Microprocessor", Serial No. 08/146,382, filed October 29, 1993 by Witt, et al. These documents are incorporated herein by reference in their entirety.

Reservation station units 210A-210D are provided to temporarily store instruction information to be speculatively executed by the corresponding functional units 212A-212D. As stated previously, each reservation station unit 210A-210D may store instruction information for up to three pending instructions. Each of the four reservation stations 210A-210D contain locations to store bit-encoded execution instructions to be speculatively executed by the corresponding functional unit and the values of operands. If a particular operand is not available, a tag for that operand is provided from reorder buffer 216 and is stored within the corresponding reservation station until the result has been generated (i.e., by completion of the execution of a previous instruction). It is noted that when an instruction is executed by one of the functional units 212A-212D, the result of that instruction is passed directly to any reservation station units 210A-210D that are waiting for that result at the same time the result is passed to update reorder buffer 216 (this technique is commonly referred to as "result forwarding"). Instructions are issued to functional units for execution after the values of any required operand(s) are made available. That is, if an operand associated with a pending instruction within one of the reservation station units 210A-210D has been tagged with a location of a previous result value within reorder buffer 216 which corresponds to an instruction which modifies the required operand, the instruction is not issued to the corresponding functional unit 212 until the operand result for the previous instruction has been obtained. Accordingly, the order in which instructions are executed may not be the same as the order of the original program instruction sequence. Reorder buffer 216 ensures that data coherency is maintained in situations where read-after-write dependencies occur.

In one embodiment, each of the functional units 212 is configured to perform integer arithmetic operations of addition and subtraction, as well as shifts, rotates, logical operations, and branch operations. It is noted that a floating point unit (not shown) may also be employed to accommodate floating point operations.

Each of the functional units 212 also provides information regarding the execution of conditional branch instructions to the branch prediction unit 220. If a branch prediction was incorrect, branch prediction unit 220 flushes instructions subsequent to the mispredicted branch that have entered the instruction processing pipeline, and causes prefetch/predecode unit 202 to fetch the required instructions from instruction cache 204 or main memory. It is noted that in such situations, results of instructions in the original program sequence which occur after the mispredicted branch instruction are discarded, including

those which were speculatively executed and temporarily stored in load/store unit 222 and reorder buffer
216. Exemplary configurations of suitable branch prediction mechanisms are well known.

5   Results produced by functional units 212 are sent to the reorder buffer 216 if a register value is
being updated, and to the load/store unit 222 if the contents of a memory location is changed. If the result is
to be stored in a register, the reorder buffer 216 stores the result in the location reserved for the value of the
register when the instruction was decoded. As stated previously, results are also broadcast to reservation
station units 210A-210D where pending instructions may be waiting for the results of previous instruction
executions to obtain the required operand values.

10

Generally speaking, load/store unit 222 provides an interface between functional units 212A-212D
and data cache 224. In one embodiment, load/store unit 222 is configured with a load/store buffer with eight
storage locations for data and address information for pending loads or stores. Decode units 208 arbitrate for
access to the load/store unit 222. When the buffer is full, a decode unit must wait until the load/store unit
15   222 has room for the pending load or store request information. The load/store unit 222 also performs
dependency checking for load instructions against pending store instructions to ensure that data coherency is
maintained.

Data cache 224 is a high speed cache memory provided to temporarily store data being transferred
20   between load/store unit 222 and the main memory subsystem. In one embodiment, data cache 224 has a
capacity of storing up to sixteen kilobytes of data. It is understood that data cache 224 may be implemented
in a variety of specific memory configurations, including a set associative configuration.

Turning next to Figure 2, a block diagram of data cache 224 is shown including a way prediction
25   structure according to the present invention. For simplicity, only those elements of data cache 224 pertaining
to the present invention are shown. Generally speaking, the way prediction structure predicts a way of data
cache 224 in which a request address may hit, and causes the predicted way to be selected as the output data
bytes of data cache 224. Therefore, the tag comparisons typically required by an associative data cache are
bypassed in performing the way selection. The data may be available at the output of the data cache after a
30   time delay from the start of the access similar to the time delay of a direct-mapped cache. If the way
prediction is correct, single cycle access of an associative cache is achieved, and the clock cycle may be
consistent with a direct-mapped cache access. If the way prediction is incorrect but the request address hits
in another way, then a subsequent clock cycle is used to fetch the correct data and to update the way
prediction to the hitting way. If the request address is a miss, then the way prediction is left unchanged and
35   the miss is processed.

Each clock cycle, load/store unit 222 may convey one or more cache access requests on an address
request bus 250 to data cache 224. In one embodiment, load/store unit 222 is configured to convey up to two
requests per clock cycle. Each request includes an address and request identification information including

an indication of whether the request is associated with a load or a store instruction. Address request bus 250 is coupled to input ports on a way prediction array 251, a cache tag array 252, and a cache data array 253. Way prediction array 251 is configured to store way predictions for data cache 224. Cache tag array 252 stores tags for the data stored in data cache 224. The data bytes are stored in cache data array 253. A cache

5   control unit 254 is coupled to cache tag array 252 and cache data array 253. Cache control unit 254 is configured to provide cache control functions, such as managing the replacement algorithm for data cache 224 and creating byte enables for store cache accesses.

As shown in Figure 2, cache tag array 252 and cache data array 253 are N-way set associative.

10   Each clock cycle, N sets of data bytes are conveyed from output ports of cache data array 253 on a set of way data buses 255A-255N (collectively referred to as way data buses 255). Each of way data buses 255 conveys data bytes from a particular way indexed by addresses conveyed on address request bus 250. One set of data bytes is conveyed on each of way data buses 255 for each address which may be conveyed on address request bus 250 in a given clock cycle. In one embodiment, data cache 224 is eight-way set associative.

15

Way data buses 255 are coupled to an N-to-One way selection block 256. In one embodiment, way selection block 256 is a set of multiplexors (one for each address which may be conveyed on address request bus 250). Each multiplexor is coupled to way data buses 255 and to a way prediction bus 257 from way prediction array 251. A way prediction for each requested address is conveyed from output ports of way

20   prediction array 251 on way prediction bus 257 to way selection block 256. The way prediction is indicative of one of the N ways of data cache 224. In one embodiment, a way prediction comprises a decoded value wherein a particular way is indicated through the setting of a particular bit within the decoded value. In another embodiment, the value is encoded as a number between zero and N-1 wherein each value corresponds to the selected way. The data bytes associated with the predicted way for each request address

25   are conveyed on a data bus 258 to load/store unit 222 and reservation stations 210.

Way prediction array 251 is configured to store way predictions for data cache 224. While cache tag array 252 and cache data array 253 are configured similarly with respect to numbers of ways and numbers of sets, way prediction array may be configured differently. In particular, more than one way prediction for a

30   particular set of cache tag array 252 and cache data array 253 may be stored within way prediction array 251. For example, if way prediction array 251 is configured with twice as many sets as cache tag array 252 and cache data array 253, then two way predictions will be stored within way prediction array 251 for each set of cache tag array 252 and cache data array 253. The number of bits of the request address used to index way prediction array 251 would be one larger than the number used to index cache tag array 252 and cache data

35   array 253 for this example, and the extra bit determines which of the two stored way predictions is selected as the way prediction for that request address. It is noted that the likelihood of storing a correct way prediction for a given address is increased by the number of way predictions stored for the indexed set of cache tag array 252 and cache data array 253. Performance may therefore be increased by using a way prediction array having more than one way prediction per set of cache tag array 252 and cache data array

253. In one embodiment, way prediction array 251 is a linear array of 512 way predictions, and each way prediction is a decoded value as described above. Generally speaking, a preferred embodiment of way prediction array 251 is a linear array comprising a number of way prediction storage locations equal in number to the number of sets multiplied by the number of ways in data cache array 253.

5

While a request address is indexing into cache tag array 252 and cache data array 253, the request address is also indexing into way prediction array 251. Way prediction array 251 conveys the indexed way prediction on way prediction bus 257, causing the data bytes associated with the predicted way to be conveyed on data bus 258. Therefore, the time delay from the beginning of an access to data being provided

10   by data cache 224 is characterized by the time delay of a direct-mapped cache access (way prediction array 251) followed by a small delay associated with the selection of output bytes using the predicted way. Such a time delay is substantially similar to the access time of a direct-mapped cache. Advantageously, associative data cache 224 is characterized by a direct-mapped cache access time.

15   Additionally, way prediction bus 257 is coupled to a tag comparison block 259. Tag comparison block 259 embodies logic circuits configured to determine whether or not the request address hits in the predicted way, hits in an unpredicted way, or misses. Tag comparison block 259 compares the indexed tags (as conveyed on a set of way tag buses 264A-264N similar to way data buses 255) to the request address, then determines if the actual hit way from the tag comparisons matches the predicted way from way

20   prediction array 251. Since the access to cache tag array 252 and the tag comparisons that follow may not complete in a given clock cycle, tag comparison block 259 produces signals early in a clock cycle subsequent to the clock cycle in which the associated addresses access data cache 224.

If a hit in the predicted way is detected for a load request, a signal is asserted on a hit predicted way

25   bus 260. Hit predicted way bus 260 is coupled between tag comparison block 259 and load/store unit 222 and reservation stations 210. A signal associated with each possible request address is conveyed on hit predicted way bus 260, and both load/store unit 222 and reservation stations 210 are configured to associate each signal with the appropriate address request from the previous clock cycle. A hit in the predicted way indicates that the data conveyed on data bus 258 in the previous clock cycle is correct. Reservation stations

30   210 may then discard any instructions which received the data and executed in the current clock cycle, and load/store unit 222 may discard the load instruction associated with the requested address. Additionally, a functional unit 212 which received the incorrect data sends a cancel indication along with the value it calculates using the incorrect data during this clock cycle such that reorder buffer 216 does not record the incorrect value and such that reservation stations 210 ignore the forwarded valued.

35

If a hit in an unpredicted way is detected for a load request, a corresponding signal is asserted on a hit unpredicted way bus 261 which is coupled to load/store unit 222 and reservation stations 210. A hit in an unpredicted way causes load/store unit 222 to cancel a request address conveyed on the portion of request address bus 250 upon which the unpredicted way hit address was conveyed in the previous clock cycle. Data

11

cache 224 uses the idle port to convey the correct data on data bus 258. The way prediction is corrected while the corrected data is conveyed on data bus 258. The way prediction is corrected via a way prediction control unit 265. Way prediction control unit 265 receives a hit way bus 263 and hit unpredicted way bus 261. When a hit in an unpredicted way signal is asserted, the way in which the hit occurs is conveyed on hit

5   way bus 263. Way prediction control unit 265 conveys the hitting way to a write port on way prediction array 251, and signals way prediction array 251 to write the value into the location indexed by the associated address stored by data cache 224 from the previous clock cycle. Thus, a hit in an unpredicted way is penalized only a clock cycle. Additionally, reservation stations 210 detect the hit in the unpredicted way and cause the data forwarded in the previous clock cycle with respect to the associated address to be invalidated.

10   If an instruction which received the incorrect data as an operand is currently being executed by the respective functional unit 212, reservations stations 210 assert a signal to other reservation stations and reorder buffer 216 indicative of the incorrect data so that the other reservation stations and reorder buffer 216 ignore any results from executing an instruction with the incorrect data.

15   If a miss is detected, a corresponding signal is asserted on a miss bus 262 which is coupled to load/store unit 222 and reservation stations 210. Load/store unit 222 records the miss for later miss processing, and reservation stations 210 perform similar actions to the case of a hit in an unpredicted way described above. Way prediction array 251 is not modified for this case, since there is no correct predicted way for a cache miss. In another embodiment, miss bus 262 is not included and a miss is detected via the

20   absence of appropriate signal assertions upon hit predicted way bus 260 and hit unpredicted way bus 261.

It is noted that the above discussion applies to the effects of way prediction on load accesses to data cache 224. Store requests may be performed according to the disclosure in a commonly assigned, co-pending patent application entitled: "A Data Cache Capable of Performing Store Accesses in a Single Clock

25   Cycle", filed concurrently herewith by Witt, et al. The disclosure of this patent application is incorporated herein by reference in its entirety. Otherwise, store requests are performed as two accesses in subsequent clock cycles. During a first access, the way of data cache 224 storing bytes associated with the store address is determined. The way prediction structure returns the predicted way. During a second access, the store is performed to the way determined in the first access. If the predicted way is found to be incorrect, the second

30   access is cancelled and the correct way is recorded by load/store unit 222 during a repeat of the first access (similar to the load access hit in an unpredicted way, described above). During a clock cycle subsequent to the repeated first access, the second access is performed. Cache control unit 254 is configured to generate byte enables for data cache 224 such that only the bytes affected by the store request are overwritten within the selected storage location. Other bytes are left unchanged.

35

In one embodiment, way prediction array 251 is initialized at power up of microprocessor 200 such that way zero is the way prediction stored in each entry. Subsequent accesses to data cache 224 cause the various stored way predictions to change as described above.

Turning now to Figure 3, a timing diagram showing the operation of the way prediction structure for two successive correct way predictions is shown. Three clock cycles are shown, as indicated by the CLK1, CLK2, and CLK3 labels shown in Figure 3. During CLK1, a request for an address "A" is conveyed from load/store unit 222 to data cache 224, as indicated by block 300. Near the end of CLK1, data from the predicted way is forwarded on data bus 258, as indicated by arrow 301. Early in CLK2, tag comparison block 259 determines that address A hits in the predicted way, and therefore the data conveyed on data bus 258 during CLK1 is correct. The corresponding signal on hit predicted way bus 260 is asserted, as indicated by arrow 302. Load/store unit 222 conveys a request for an address "B" in CLK2, as indicated by block 303. Data from the predicted way is forwarded on data bus 258 as indicated by arrow 304. During CLK3, tag comparison block 259 determines that address B hits in the way predicted during CLK2. Once again, the corresponding signal on hit predicted way bus 260 is asserted (as indicated by arrow 305). As can be seen from Figure 3, the way prediction structure enables single clock cycle access to associative data cache 224 with access times characteristic of direct-mapped caches as long as the way prediction is correct.

Turning next to Figure 4, a timing diagram showing two consecutive accesses in which the first access hits in an unpredicted way is shown. As with Figure 3, consecutive clock cycles are depicted on the timing diagram. The clock cycles are labeled CLK1, CLK2, CLK3, and CLK4. During CLK1, load/store unit 222 conveys a request for an address "A" to data cache 224, as indicated by block 400. Near the end of CLK1, data from the predicted way is forwarded on data bus 258, as indicated by arrow 401. Early in CLK2, tag comparison block 259 detects a hit in one of the unpredicted ways of data cache 224, and therefore the data conveyed on data bus 258 during CLK1 is incorrect. The corresponding signal on hit unpredicted way bus 261 is asserted, as indicated by arrow 402. Load/store unit 222 begins conveying a request for an address "B" on address request bus 250, but cancels that request due to the hit in the unpredicted way, as indicated by block 403. Address A is subsequently conveyed on address request bus 250 (shown as block 404). Near the end of CLK2, the way prediction array is updated with the corrected way prediction for address A and the correct data is forwarded on data bus 258 (as indicated by arrow 405). During CLK3, a hit in the predicted way will be signaled since the corrected way is conveyed on way prediction bus 257, as indicated by arrow 406. Therefore, one extra clock cycle is used to provide data which hits in an unpredicted way.

Also during CLK3, load/store unit 222 conveys the request for address B that it attempted to convey in CLK2 (as indicated by block 407). Since a hit in the predicted way is signaled by data cache 224, the request for address B remains active in CLK3 and data from the predicted way for address B is forwarded (as indicated by arrow 408). In CLK4, arrow 409 indicates that address B hits in the predicted way and so data forwarded during CLK3 with respect to address B data is correct. Another address request could be conveyed by load/store unit 222 in CLK4 and it would receive data forwarded from its predicted way near the end of CLK4.

13

It is noted that although the foregoing discussion describes the way prediction structure with respect to a data cache, a similar structure is contemplated for instruction caches as well.

In accordance with the above disclosure, a way prediction structure has been described which
5   enables access to an associative cache with an access time which is characteristic of a direct-mapped cache. Such a cache and way prediction structure may be advantageously incorporated into a high frequency superscalar microprocessor. Performance may be increased due to the better hit rates available with an associative cache and due to the shorter clock cycle achievable with the way prediction structure.

10   Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.

WHAT IS CLAIMED IS:

1. A way prediction structure for a cache comprising:

5

a way prediction array including a plurality of storage locations wherein each one of said plurality

of storage locations is configured to store a way value, and wherein said way prediction

array is configured to receive at least one request address; and

10

a way prediction control unit coupled to said way prediction array wherein said way prediction

control unit is configured to update said way prediction array.

2. The way prediction structure as recited in claim 1 wherein said way prediction array is a linear array.

15   3. The way prediction structure as recited in claim 1 wherein said way prediction array is configured to

convey a way prediction to said cache.

4. The way prediction structure as recited in claim 3 wherein said way prediction is a particular way value

stored within a selected one of said plurality of storage locations.

20

5. The way prediction structure as recited in claim 4 wherein said selected one of said plurality of storage

locations is indexed by said request address.

6. The way prediction structure as recited in claim 3 wherein said way prediction control unit is configured

25   to update said way prediction array during a clock cycle in which said way prediction is found to be

incorrect.

7. The way prediction structure as recited in claim 3 further comprising a tag comparison block adapted to

receive a plurality of tag addresses from said cache, said request address, and said way prediction, and

30   wherein said tag comparison block is coupled to said way prediction array and to said way prediction control

unit, and wherein said tag comparison block is configured to convey a plurality of signals indicative of the

correctness of said way prediction.

8. The way prediction structure as recited in claim 7 wherein said plurality of signals includes a hit predicted

35   way signal indicative of a hit in said cache in a way predicted by said way prediction structure.

9. The way prediction structure as recited in claim 7 wherein said plurality of signals includes a hit

unpredicted way signal indicative of a hit in said cache in a way not predicted by said way prediction

structure.

10. The way prediction structure as recited in claim 9 wherein said way prediction control unit is adapted to receive said hit unpredicted way signal and wherein said way prediction control unit determines that said way prediction is incorrect if said hit unpredicted way signal is asserted.

5

11. The way prediction structure as recited in claim 7 wherein said plurality of signals includes a hit way signal indicative of a way of said cache in which said request address hits.

12. The way prediction structure as recited in claim 11 wherein said way prediction control unit is adapted to

10   receive said hit way signal and wherein said way prediction control unit is configured to update said way prediction array with said way during a clock cycle in which said way prediction if found to be incorrect.

13. The way prediction structure as recited in claim 1 further comprising said cache.

15   14. The way prediction structure as recited in claim 13 wherein said cache is a data cache.

15. The way prediction structure as recited in claim 14 further comprising a load/store unit coupled to said data cache wherein said load/store unit is configured to execute load and store instructions.

20   16. The way prediction structure as recited in claim 15 wherein said load/store unit is configured to access said data cache with said request address during a second clock cycle.

17. A superscalar microprocessor comprising a data cache including:

25   a way prediction array including a plurality of storage locations wherein each one of said plurality
                  of storage locations is configured to store a way value, and wherein said way prediction
                  array is configured to receive at least one request address;

           a way prediction control unit coupled to said way prediction array wherein said way prediction

30   control unit is configured to update said way prediction array; and

           a second plurality of storage locations configured to store data bytes and to receive said request
                  address, wherein said second plurality of storage locations is coupled to said way
                  prediction array, and wherein one of said second plurality of storage locations is selected

35   by said way value to convey data bytes upon an output bus.

18. The superscalar microprocessor as recited in claim 17 wherein said way prediction array is a linear array.

19. The superscalar microprocessor as recited in claim 17 wherein said second plurality of storage locations

is an associative array.

20. The superscalar microprocessor as recited in claim 17 wherein said way prediction control unit is configured to update said way prediction array during a clock cycle in which said way prediction is found to be incorrect.

21. The superscalar microprocessor as recited in claim 17 wherein said data cache further includes a third plurality of storage locations and a tag comparison block adapted to receive a plurality of tag addresses from said third plurality of storage locations, and wherein said tag comparison block is configured to receive said request address and said way prediction, and wherein said tag comparison block is coupled to said way prediction array and to said way prediction control unit, and wherein said tag comparison block is configured to convey a plurality of signals indicative of the correctness of said way prediction.

22. The superscalar microprocessor as recited in claim 21 further comprising a load/store unit coupled to said data cache wherein said load/store unit is configured to execute load and store memory accesses.

23. The superscalar microprocessor as recited in claim 22 wherein said load/store unit is configured to access said data cache with said request address during a second clock cycle.

24. The superscalar microprocessor as recited in claim 22 further comprising a plurality of functional units coupled to said load/store unit, wherein said plurality of functional units are configured to calculate addresses for said load/store unit such that said load/store unit may perform said load and store memory accesses.

25. The superscalar microprocessor as recited in claim 24 wherein said plurality of functional units are further configured to execute decoded instructions.

26. The superscalar microprocessor as recited in claim 25 further comprising a plurality of reservation stations configured to store said decoded instructions prior to execution by said plurality of functional units, wherein each of said plurality of reservation stations is coupled to a respective one of said plurality of functional units.

27. The superscalar microprocessor as recited in claim 26 further comprising a plurality of decode units configured to decode instructions into said decoded instructions, wherein each of said plurality of decode units is coupled to a respective one of said plurality of reservation stations.

28. The superscalar microprocessor as recited in claim 27 further comprising:

  an instruction cache for storing said instructions; and

17

an instruction alignment unit coupled between said instruction cache and said plurality of decode units, wherein said instruction alignment unit is configured to align said instructions from said instruction cache to said plurality of decode units.

5   29. The superscalar microprocessor as recited in claim 28 further comprising a reorder buffer coupled to said plurality of decode units and to said plurality of functional units, wherein said reorder buffer is configured to store speculative register states.

    30. The superscalar microprocessor as recited in claim 29 further comprising a register file for storing non-
10 speculative register states, wherein said register file is coupled to said reorder buffer, and wherein said reorder buffer is configured to store said speculative register states to said register file in program order.

    31. A method for accessing an associative cache with a direct-mapped access time comprising:

15   predicting a way of said associative cache that an access will hit; and

        validating said prediction during a subsequent clock cycle.

    32. The method as recited in claim 31 wherein said predicting step is performed by accessing a way
20 prediction array.

    33. The method as recited in claim 32 further comprising updating said way prediction array if said validating indicates that said predicting is incorrect.

25 34. The method as recited in claim 31 further comprising cancelling incorrect data from an access performed according to said predicting.

    35. The method as recited in claim 34 further comprising ignoring said incorrect data when said cancelling is received.
30
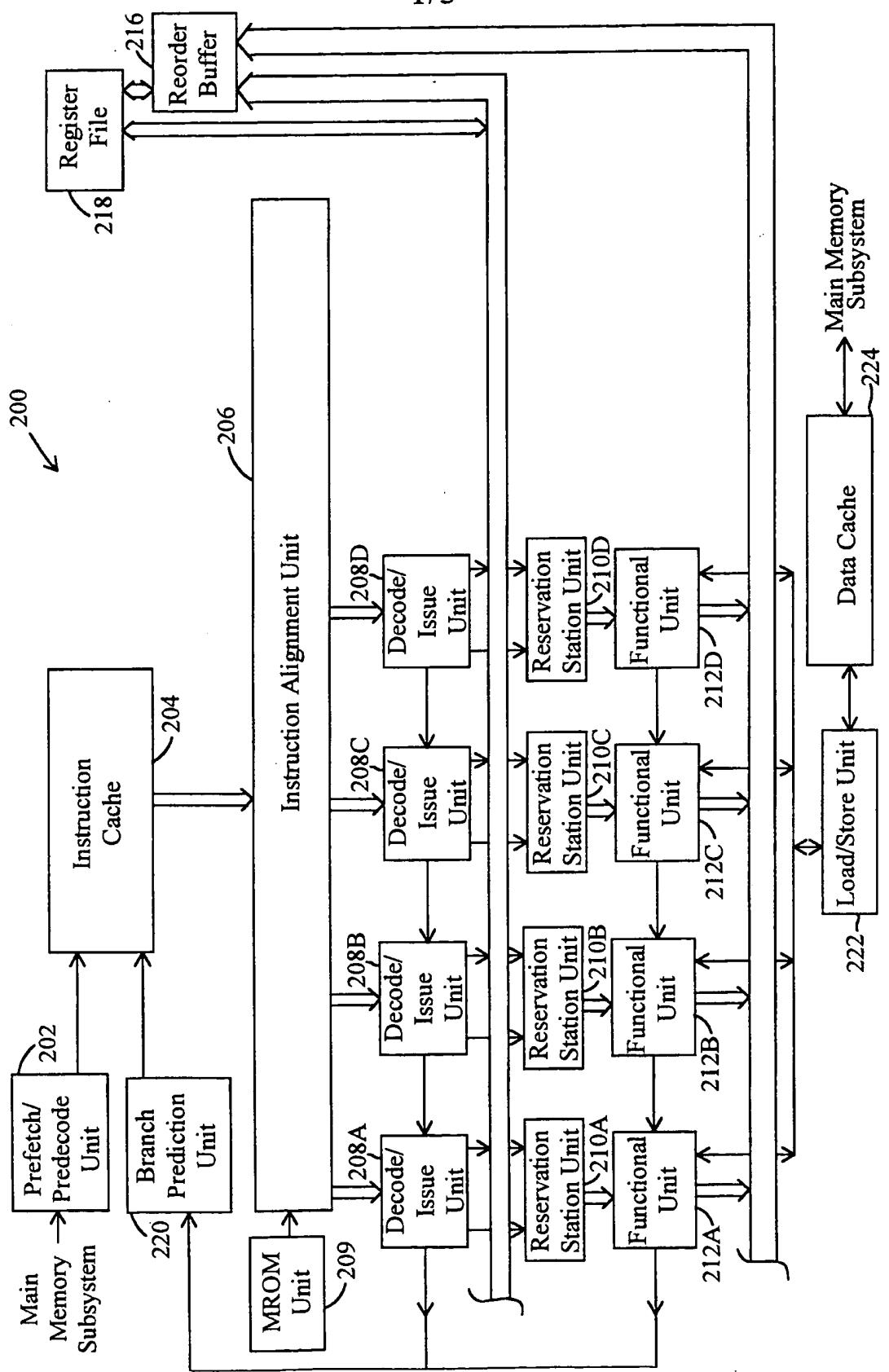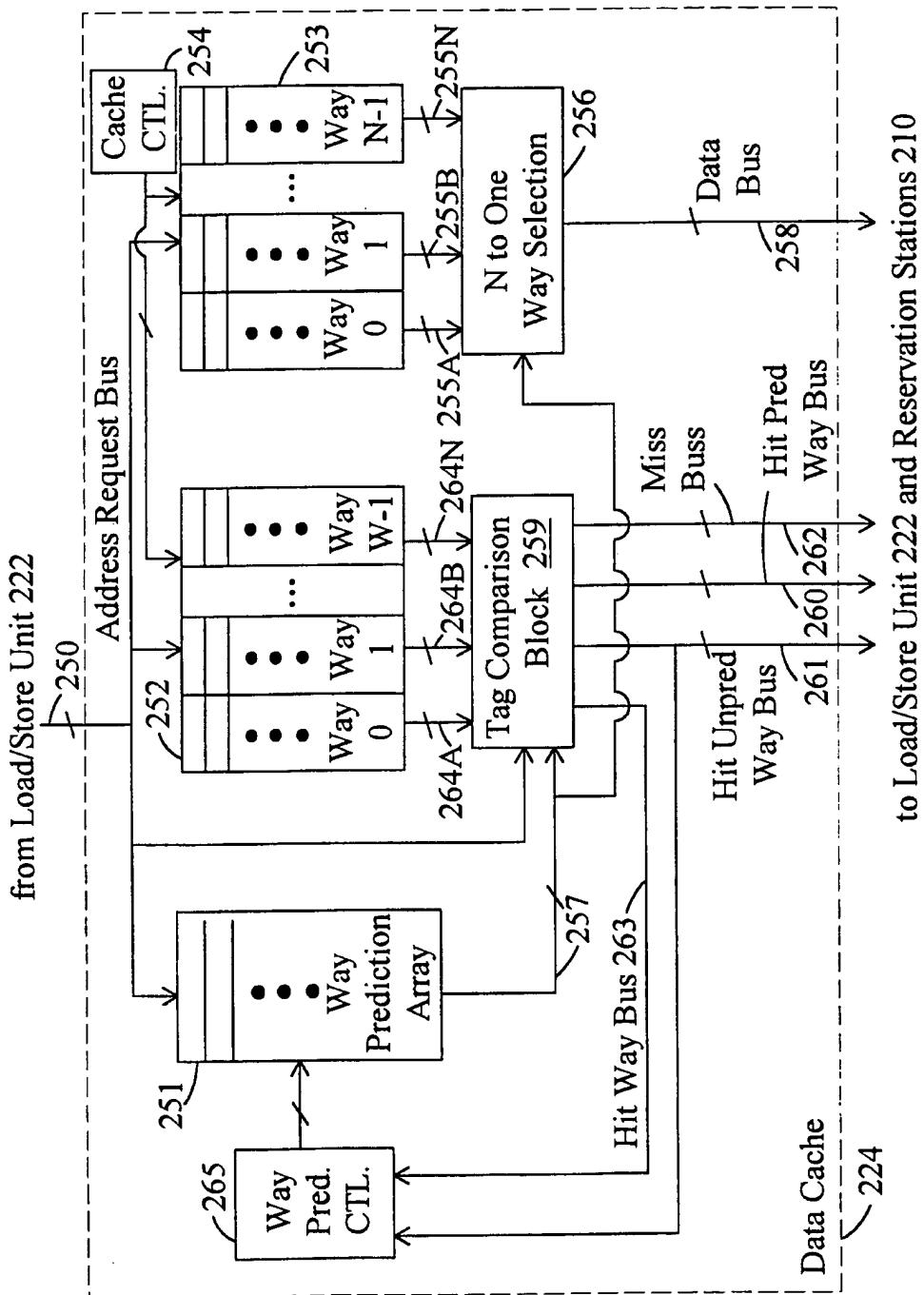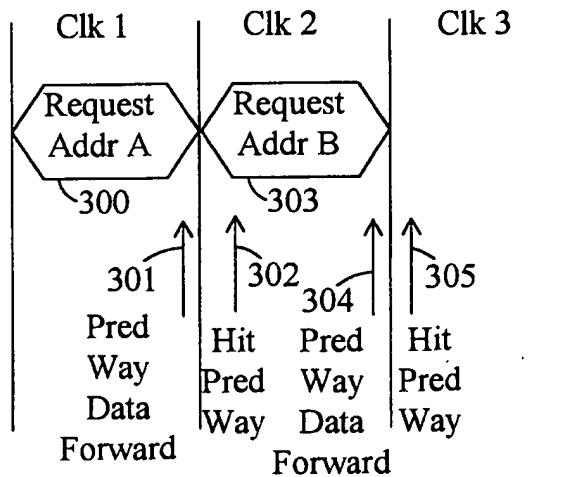
Fig. 1

2/3

Fig. 2

3/3

Clk 1   Clk 2   Clk 3

Request   Request
Addr A   Addr B
⤷300   ⤷303

   301   302 304   305
Pred   Hit   Pred   Hit
Way   Pred Way   Pred
Data   Way   Data   Way
Forward   Forward

Fig. 3

Clk 1   Clk 2   Clk 3   Clk 4

Request   B   Addr A   Request
Addr A   Addr B
⤷400   403 ⤷404   ⤷407

   401   402   406 408   409
          405
Pred   Correct Hit   Pred   Hit
Way   Hit   Way   Pred Way   Pred
Data   Unpred Pred   Way   Data   Way
Forward   Way   Forward
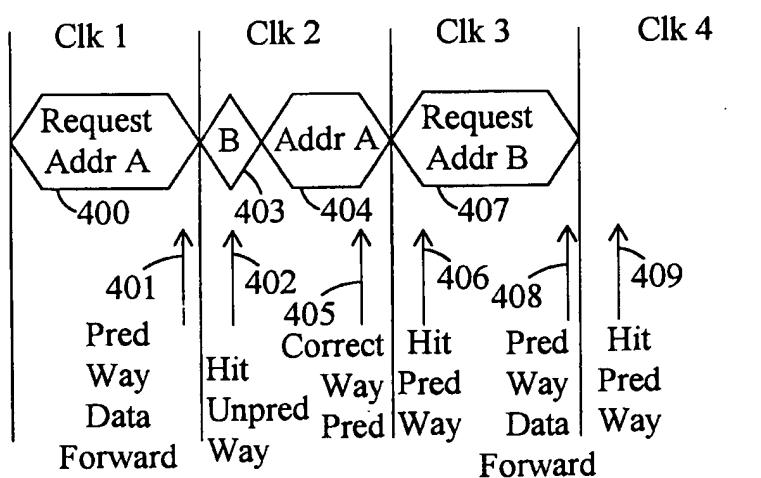
Fig. 4

INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6   G06F12/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 6   G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 235 697 A (STEELY JR SIMON C ET AL) 10 August 1993 see column 3 - column 11 ---	1-23, 31-35
X	EP 0 675 443 A (DIGITAL EQUIPMENT CORP) 4 October 1995	1-23, 31-35
Y	see column 2, line 43 - column 8, line 7; figures 2,3 ---	24-26
Y	EP 0 651 321 A (ADVANCED MICRO DEVICES INC) 3 May 1995 cited in the application	24-26
A	see page 10, line 29 - page 20, line 46; figures 3,6 ---	27-30
	-/--	

X	Further documents are listed in the continuation of box C.		X	Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 June 1997	2 5. 06. 97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Nielsen, O

Form PCT/ISA/210 (second sheet) (July 1992)

1

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 467 152 A (HITACHI LTD) 22 January 1992 see abstract see column 8, line 24 - line 27	26,27

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
US 5235697 A	10-08-93	NONE		
EP 0675443 A	04-10-95	NONE		
EP 0651321 A	03-05-95	JP	7182160 A	21-07-95
		US	5574928 A	12-11-96
EP 0467152 A	22-01-92	JP	4077925 A	12-03-92
		US	5408625 A	18-04-95